(19) RÉPUBLIQUE FRANÇAISE

INSTITUT NATIONAL
DE LA PROPRIÉTÉ INDUSTRIELLE

PARIS

(11) N° de publication : **2 806 505**
(à n'utiliser que pour les commandes de reproduction)

(21) N° d'enregistrement national : **00 03498**

(51) Int Cl[7] : **G 06 K 19/067**

(12) DEMANDE DE BREVET D'INVENTION A1

(22) **Date de dépôt :** 15.03.00.

(30) **Priorité :**

(43) **Date de mise à la disposition du public de la demande :** 21.09.01 Bulletin 01/38.

(56) **Liste des documents cités dans le rapport de recherche préliminaire :** *Se reporter à la fin du présent fascicule*

(60) **Références à d'autres documents nationaux apparentés :**

(71) **Demandeur(s) :** *SCHLUMBERGER SYSTEMES* *Société anonyme* — FR.

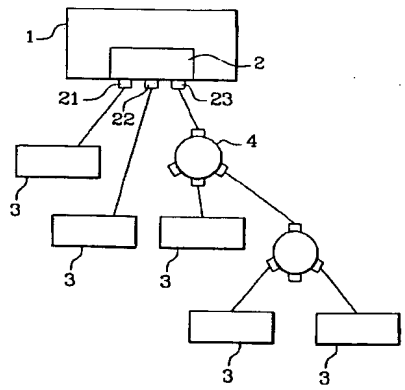
(72) **Inventeur(s) :** GELZE MATHIAS et DRABCZUK NICOLAS.

(73) **Titulaire(s) :**

(74) **Mandataire(s) :** SCHLUMBERGER SYSTEMES.

(54) PROCEDE DE COMMUNICATION ENTRE UNE CARTE A PUCE ET UNE STATION HOTE.

(57) L'invention concerne un procédé de communication entre, d'une part, une station hôte telle qu'un ordinateur personnel et, d'autre part, un objet portatif à microcontrôleur tel qu'une carte à puce, ledit objet portatif étant connecté, par un système de bus, à ladite station hôte. L'invention se caractérise en ce que le procédé comporte une étape selon laquelle une requête spécifique est communiquée à l'objet portatif à microcontrôleur par la station hôte. L'invention s'applique en particulier à des cartes à puce susceptibles de communiquer directement avec un ordinateur hôte selon le mode transfert de contrôle de la norme USB.



FR 2 806 505 - A1

PROCEDE DE COMMUNICATION ENTRE UNE CARTE A PUCE ET UNE STATION HOTE

Les cartes à puce sont des objets portatifs normalisés définis dans la norme ISO 7816, qui permettent notamment d'assurer une gestion sécurisée de données confidentielles et d'identification. De manière à communiquer avec le monde extérieur, ces cartes utilisent généralement des protocoles de communication définis dans les parties troisième et quatrième de la norme précitée. Il s'agit en particulier du protocole bien connu de l'homme du métier sous la dénomination T=0 (t égal à zéro), qui met en œuvre des commandes d'un format défini : les commandes APDU (Application Protocol Data Unit).

La norme USB (Universal Serial Bus), qui décrit un système de bus série universel, a été développée pour assurer une gestion simple et rapide des échanges de données entre une station hôte, par exemple un station de travail formée par un ordinateur personnel, et un dispositif périphérique quelconque, par exemple une imprimante ou un clavier. L'utilisation de ce système présente de nombreux avantages. Tout d'abord, il nécessite deux lignes conductrices, V_{BUS} et GND, pour l'alimentation du dispositif périphérique et deux lignes conductrices, D+ et D-, pour une transmission différentielle des signaux de données. D'autre part, il autorise des vitesses de transmission des données généralement supérieures à celles proposées par les liaisons séries classiquement installées sur les ordinateurs personnels. Ces vitesses sont de 12 Mb/s (Méga bits par seconde) en ce qui concerne la vitesse pleine et de 1,5 Mb/s en ce qui concerne la vitesse basse. Ensuite, il permet un «Plug & Play» à chaud des dispositifs périphériques, c'est-à-dire une reconnaissance dynamique desdits dispositifs par l'ordinateur hôte. Grâce à cette reconnaissance, les pilotes des dispositifs périphériques, qui résident dans une mémoire de masse de l'ordinateur hôte, sont chargés dans une mémoire vive dudit ordinateur uniquement à la connexion desdits périphériques. Ces mêmes pilotes sont déchargés de ladite mémoire vive à la déconnexion des

périphériques. En outre, le système de bus série universel rend possible une connexion jusqu'à 126 dispositifs périphériques en cascade sur un unique port physique USB. Enfin, les périphériques USB ne monopolisent pas d'interruption matérielle IRQ (Interrupt ReQuest) gérée les composants de l'ordinateur.

Aujourd'hui, le besoin de sécurisation de l'accès aux stations hôte ainsi que de l'accès aux serveurs associés auxdites stations est grandissant. Il en va de même pour le besoin de sécurisation des transferts de données gérées à partir de telles stations, en particulier à partir de logiciels applicatifs desdites stations dédiés notamment aux mèls ou à la navigation sur l'Internet, et pour lesquels on souhaiterait une authentification des données au moyen d'algorithmes de cryptage permettant de certifier lesdites données et de les signer.

Compte tenu de l'état de la technique préalablement exposé, on a naturellement répondu aux besoins de sécurisation précités en utilisant des cartes à puce, fonctionnant selon les protocoles organisés par les parties troisième et quatrième de la norme ISO 7816, avec des lecteurs de cartes à puce spécifiques, connectés aux ports USB d'un ordinateur hôte et effectuant une conversion de protocole USB/ISO. De tels lecteurs communiquent, d'une part, avec l'ordinateur hôte selon le système USB et, d'autre part, avec la carte, selon le système ISO.

Toutefois, ces lecteurs sont très coûteux. En effet, ils doivent comporter des moyens pour générer une horloge destinée à cadencer le fonctionnement de l'unité centrale de traitement CPU (Central Processing Unit) du micro-contrôleur de la carte via la plage de contact Clock (CLK) de la carte. Ils doivent en outre comporter des moyens pour générer un signal de remise à zéro et pour transmettre ce signal à ladite carte, via une plage de contact spécifique de celle-ci, la plage dite de Reset (RST).

De plus, la carte étant finalement une carte ISO pure, les procédés de communication avec cette carte ne montrent pas les avantages précités du

système USB relatifs en particulier aux lignes conductrices en nombre limité et aux débits importants de données.

Compte tenu de ce qui précède, un problème que se propose de résoudre l'invention est de réaliser un procédé de communication permettant de connecter un objet portatif tel qu'une carte à puce à moindres coûts à une station de travail telle qu'un ordinateur personnel selon le système USB.

La solution proposée de l'invention à ce problème posé a pour premier objet un procédé de communication entre, d'une part, une station hôte telle q'un ordinateur personnel et, d'autre part, un objet portatif à microcontrôleur tel qu'une carte à puce, ledit objet portatif étant connecté, par un système de bus, à ladite station hôte, caractérisé en ce qu'il comporte une étape selon laquelle une requête spécifique est communiquée à l'objet portatif à microcontrôleur par la station hôte.

De manière avantageuse, le système de bus est un système de bus série universel USB et en ce que la requête spécifique est communiquée à l'objet portatif à microcontrôleur selon le mode transfert de contrôle dudit système ; la requête spécifique est une requête spécifique qui assure une fonctionnalité d'un lecteur d'objet portatif à microcontrôleur ; le microcontrôleur comporte un ensemble associant une unité centrale de traitement ainsi qu'une mémoire volatile et en ce que la requête spécifique est une requête (DoReset()) qui déclenche une remise à zéro de la mémoire volatile dudit ensemble ; la requête spécifique est une requête (GetATR()) qui permet de récupérer une chaîne de réponse à une remise à zéro de l'objet portatif ; la requête spécifique est une requête (SendAPDU()) qui permet à la station hôte d'envoyer à l'objet portatif une entête d'une commande ; la requête spécifique est une requête spécifique (GetData()) qui permet à la station hôte de récupérer des données envoyées par l'objet portatif et de récupérer un mot d'état ; la requête spécifique est une requête (SendData()) qui permet à la station hôte de communiquer des données à l'objet portatif ; la requête spécifique est une requête (IsReady()) qui permet d'éviter un déclenchement, par la station hôte, d'un mode de

consommation de courant électrique réduit chez l'objet portatif ; l'objet portatif communique une réponse (OS-STATUS) à la station hôte en réponse à la requête permettant le déclenchement, par ladite station, d'un mode de consommation de courant électrique réduit chez l'objet portatif, cette réponse étant codée de manière à définir un état courant de l'objet portatif ; l'état courant de l'objet portatif est un état de mutisme ou un état selon lequel la carte est en cours de traitement ; l'objet portatif à microcontrôleur est une carte à microcontrôleur ; le microcontrôleur de la carte comporte une mémoire non volatile qui comprend un système d'exploitation susceptible de communiquer selon un protocole mettant en œuvre des commandes APDU telle que définies dans la norme ISO7816.

Par ailleurs, la solution proposée de l'invention à ce problème posé a pour second objet un objet portatif à microcontrôleur tel qu'une carte à puce, apte à communiquer avec une station hôte telle qu'un ordinateur personnel, par un système de bus connecté, d'une part, audit objet portatif et, d'autre part, à ladite station hôte, caractérisé en ce que l'objet portatif est apte à communiquer avec la station hôte, directement.

De manière avantageuse, l'objet portatif est constitué par une carte à puce ; le système de bus est un système de bus USB ; l'objet portatif comporte un ensemble associant une unité centrale de traitement et une mémoire non volatile embarquant un système d'exploitation apte à assurer une gestion de commandes APDU telle que définies dans la norme ISO7816.

L'invention sera mieux comprise à la lecture de l'exposé non limitatif qui va suivre. Cet exposé doit être lu au regard des dessins annexés, dans lesquels :

- la figure 1 représente des schémas de connexion possibles entre une station de travail hôte et un objet portatif selon l'invention ;

- la figure 2 montre un mode de connexion d'un ordinateur personnel hôte à une carte à puce selon l'invention ;

- la figure 3 montre, en perspective, un élément d'un connecteur apte à recevoir une carte à puce pour une connexion à un ordinateur hôte selon l'invention ;

- la figure 4 montre, en vue de face agrandie, les contacts d'une carte à puce destinée à une connexion à un ordinateur hôte selon l'invention ;

- la figure 5 schématise différents éléments intervenant dans le fonctionnement d'un microcontrôleur d'une carte destinée à une connexion à un ordinateur hôte selon l'invention ;

- la figure 6 schématise une architecture logique d'un système de communication entre une carte et une application logicielle d'un ordinateur hôte selon l'invention ;

- la figure 7 montre le déroulement d'une cession de communication avec une carte à puce selon l'invention ;

- les figures 8A et 8B montrent des transactions qui ont lieu dans un mode d'exécution d'une commande de type ISO 1 par la carte ;

- les figures 9A à 9D montrent des transactions qui ont lieu dans un mode d'exécution d'une commande de type ISO 2 par la carte ; et

- les figures 10A à 10D montrent des transactions qui ont lieu dans un mode d'exécution d'une commande de type ISO 3 par la carte.

L'invention s'applique en particulier dans le cadre de la sécurisation d'une station hôte, par exemple munie d'un système d'exploitation distribué par la société Microsoft sous la dénomination Windows 2000 protégée en tant que marque. Ce système d'exploitation, ainsi que certaines applications logicielles conçues pour ce système d'exploitation, prévoient l'utilisation d'une carte destinée notamment à la sécurisation de transferts de données, par exemple à la signature de méls, et à la sécurisation d'accès aux réseaux informatiques, par exemple au moyen d'algorithmes d'authentification ou de non-répudiation. De manière générale, l'invention peut être mise en œuvre sur toutes les cartes dont les systèmes d'exploitation sont compatibles avec les parties troisième et quatrième de la norme ISO 7816.

A la figure 1, on a schématisé une station hôte 1 comportant un répartiteur (Hub) intégré 2, ledit répartiteur 2 étant muni de ports 21, 22 et 23 spécifiques définis dans la version 1.1 de la norme USB publiée le 23 septembre 1998. Ces ports USB peuvent être connectés à un objet portatif 3 à microcontrôleur selon l'invention, soit directement, comme c'est le cas des objets connectés aux ports 21 et 22, soit de manière indirecte, via un autre répartiteur 4, comme c'est le cas des objets connectés au port 23.

Ainsi que cela est montré à la figure 2, la station hôte est par exemple une station de travail formée par un ordinateur personnel 1 et l'objet portatif à microcontrôleur est une carte à puce 3. La carte à puce 3 est connectée à un connecteur 5 qui n'est pas un lecteur, un lecteur disposant en effet de moyens actifs pour lire et/ou écrire une carte et/ou pour permettre cette lecture et/ou cette écriture.

L'ordinateur 1 possède classiquement une unité centrale 11 reliée à un moniteur 12 et à un clavier. L'unité centrale 11 comporte une carte mère. Cette carte mère comprend en particulier un microprocesseur, des barrettes de mémoire volatile. Elle est reliée à un disque dur, qui constitue la mémoire de masse de l'ordinateur, ainsi qu'à un port USB au moins qui est compris dans un répartiteur intégré à l'ordinateur.

Si l'on se rapporte maintenant à la figure 6, il apparaît que l'ordinateur hôte 1 comporte au moins une application logicielle 13 utilisant une carte à puce. Il comporte en outre une partie logicielle PC/SC 14, qui est chargée d'assurer la gestion de l'interface utilisée par l'application 13. Il comporte encore un logiciel pilote intermédiaire 15 composé de deux parties logicielles principales non représentées sur la figure 6. Il s'agit d'une première partie chargée en mémoire vive de l'ordinateur hôte au démarrage, qui assure l'interface avec la partie logicielle PC/SC 14 et qui simule la présence d'un lecteur d'une ou plusieurs cartes à puce selon l'invention connectées à l'ordinateur hôte. C'est un lecteur virtuel. Il s'agit en outre d'une seconde partie enregistrée dans la mémoire de masse de l'ordinateur hôte, qui est chargée dans sa mémoire vive lorsqu'une

carte est connectée à l'ordinateur hôte, adressée et configurée. Cette seconde partie est chargée d'acheminer les informations issues de la partie PC/SC ou issues de la carte vers leurs destinataires respectifs et effectue une conversion des données. Il comporte par ailleurs une partie contrôleur de l'hôte 16 qui est chargée de gérer la répartition des données sur le bus USB. Il comporte enfin une partie hardware 17 qui constitue l'interface entre l'hôte et le mode extérieur.

La carte 3 montrée à la figure 2 est par exemple une carte au format ISO standard ou une carte au format dit plug-in décrit dans cette même norme ISO7816 ou dans la norme ETSI GSM 11.11. Une telle carte est représentée plus en détails à la figure 3. Elle comporte un corps 31 de carte plastique dans lequel on a inséré un module électronique comprenant un microcontrôleur connecté, par des fils de connexion, à des plages 32 de contact affleurantes à la surface dudit corps de carte.

La figure 4 représente les plages de contacts 32 de la carte 3. Elles sont par exemple au nombre de huit. Il s'agit des plages C1, C2, C3, C4, C5, C6, C7 et C8. Les plages C1 et C5 sont respectivement connectées à des plots Vcc et GND du microcontrôleur de la carte et qui servent à l'alimentation du microcontrôleur. Les plages C4 et C8 sont des plages respectivement connectées à des plots D+ et D- dudit microcontrôleur, qui constituent une paire différentielle pour une transmission de données selon le système de bus USB. Les autres plages sont utilisées pour une transmission des données selon la norme ISO, sans utilisation du système précité de bus USB.

Le microcontrôleur 33 de la carte 3 est schématisé à la figure 5. Il comporte un ensemble 331 associant une unité centrale de traitement CPU ainsi que des mémoires volatile RAM, non volatile ROM et EEPROM, la mémoire ROM embarquant le système d'exploitation de la carte. Il comporte en outre une interface de communication 332 selon le système ISO, un moteur USB 333 associé, d'une part, à un système de transmission 334 et, d'autre part, à des registres 335, ainsi qu'une External Block Interface (EBI) 336, c'est-à-dire une

interface de bloc externe. Le système de transmission est relié au moins aux plots D+ et D- de la carte. Il est en outre relié aux plots Vcc et GND pour l'alimentation. Le système d'exploitation 337 de la carte, l'EBI 336 et le moteur USB 333 sont représentés à la figure 6.

Ainsi que cela est montré à la figure 3, la carte est en pratique insérée dans un connecteur 5 de carte. Dans l'invention, ce connecteur 5 est réduit. Il possède uniquement un connecteur USB 51 et un connecteur 52 pour la carte 3.

Selon la norme USB, les données peuvent être transférées selon deux vitesses une vitesse pleine (Full speed) autorisant un débit de données à 12 Mb/s et une vitesse basse (Low speed) autorisant un débit de données à 1, 5 Mb/s. Dans l'invention, les données sont transférées à la vitesse basse. Ainsi, il est possible de générer un signal d'horloge interne à partir des lignes de données du bus USB. De ce fait, le connecteur 5 ne comporte pas de moyens pour fournir un signal d'horloge à la carte 3.

Selon la norme USB, quatre modes de transfert de données sont prévus. Les modes de transfert en vrac (Bulk transfert) et isochrone (Isochronous transfert) sont uniquement prévus pour être mis en œuvre dans une communication à vitesse pleine. Les modes de transfert de contrôle (Control transfert) et transfert avec interruption (Interrupt transfert) sont par contre prévus pour être mis en œuvre dans une communication à basse vitesse ou à pleine vitesse.

Dans l'invention, la carte forme un périphérique USB qui communique directement avec l'ordinateur hôte dans le mode transfert de contrôle. La carte est donc capable d'interpréter et de traiter des données qui lui sont adressées sous forme de signaux USB à basse vitesse, sur le bus USB. Elle dispose par ailleurs d'un programme permettant le traitement des requêtes USB propres au transfert de contrôle et en particulier des requêtes classiques, qui permettent à l'ordinateur hôte de récupérer les descripteurs de la carte, de lui attribuer une adresse et de la configurer. En effet, selon la norme USB, l'utilisation du mode

transfert de contrôle est requise pour tous les périphériques USB pour la récupération de leurs descripteurs, l'attribution de leur adresse et leur configuration. La norme USB ne suggère pas d'utiliser le mode transfert de transfert pour une gestion des transferts de données en dehors d'étapes de contrôle du type précité.

En plus des requêtes USB classiques permettant la reconnaissance, l'adressage et la configuration du périphérique, six requêtes vendeur-spécifiques (Vendor Specific) ont été définies. La carte comporte des moyens pour reconnaître et traiter ces requêtes vendeur-spécifiques. Ces requêtes vendeur-spécifiques permettent de reproduire le fonctionnement d'une carte compatible ISO 7816-3 et ISO 7816-4 associée à un lecteur actif de carte à puce, en utilisant le protocole USB et le bus de données associé et sans utiliser d'interface supplémentaire que constituerait ce lecteur. Elles sont chargées en particulier d'assurer le traitement des commandes APDU et les phases d'initialisation ou de réinitialisation du microcontrôleur de la carte sans réinitialisation de l'interface de communication avec l'ordinateur hôte.

La carte est gérée, d'une part, par le pilote installé sur l'ordinateur hôte qui est responsable de l'envoi des requêtes vendeur-spécifiques et, d'autre part, par le moteur USB contenu dans le microcontrôleur de la carte et son système d'exploitation, tous deux étant responsables de la reconnaissance et du traitement de ces mêmes requêtes.

En définitive, la carte fonctionne «comme si» elle était reliée à un lecteur de carte à puce, en utilisant le protocole USB, ce qui signifie que le changement d'interface, lecteur de carte à puce ISO vers connecteur USB, est transparent pour le niveau applicatif de l'ordinateur hôte.

Les requêtes vendeur-spécifiques selon l'invention sont définies dans le tableau ci-après. Dans ce tableau :

- Les valeurs données dans la colonne bmRequest identifie les caractéristiques des requêtes. Si la valeur du bmRequest est 40h, la requête est une requête vendeur-spécifique dont la phase de donnée est transmise de

l'hôte vers la carte. Si la valeur du bmRequest est C0h, la requête est une requête vendeur-spécifique dont la phase de donnée est transmise de la carte vers l'hôte.

- Les valeurs données dans la colonne bRequest permettent au moteur USB d'identifier les requêtes DoReset() et IsReady() en ne testant qu'un seul bit à chaque fois, les bits 4 et 5 si le bit de poids faible est le bit 0.

- Les valeurs données dans la colonne wValue sont spécifiques à la requête.

- Il en va de même des valeurs données dans la colonne wIndex.

- Les valeurs données dans la colonne wLength spécifient le nombre d'octets dans la phase de données de la requête.

- Le mode indiqué dans la dernière colonne du tableau correspond au sens de circulation des données USB. «OUT» signifie que les données circulent de l'ordinateur hôte vers la carte et «IN» signifie que les informations circulent de la carte vers l'ordinateur hôte, pendant la phase de données.

Requête	bmRequest	bRequest	wValue	wIndex	wLength	Mode
DoReset()	40h	90h	0000h	0000h	0000h	OUT
GetATR()	C0h	83h	0000h	0000h	Lgth	IN
GetData()	C0h	81h	0000h	0000h	Lgth	IN
IsReady()	C0h	A0h	0000h	0000h	0100h	IN
SendAPDU()	40h	80h	0000h	0000h	0500h	OUT
SendData()	40h	82h	0000h	0000h	Lgth	OUT

Tout d'abord, deux requêtes sont dédiées à la séquence de remise à zéro de la carte. Il s'agit des requêtes DoReset() et GetATR().

La requête DoReset() permet de déclencher une remise à zéro du microcontrôleur et de la mémoire vive RAM de l'ensemble 331 associée sans remettre à zéro l'interface de communication avec l'ordinateur hôte. Elle est traitée entièrement par le moteur USB 333 contenu dans le microcontrôleur et ne nécessite aucune intervention de la part du système d'exploitation de la carte. Le traitement automatique de la requête vendeur-spécifique DoReset()

permet à la carte à puce de gérer elle-même le signal Reset et de conserver ainsi, associée à la requête vendeur-spécifique GetATR(), le fonctionnement normal du signal Reset dans le mode ISO.

La requête GetATR() permet de récupérer la chaîne de réponse à la remise à zéro (ATR pour Answer To Reset) de la carte. Cette réponse est définie dans la norme ISO 7816-3. Elle identifie la carte.

On notera que la plupart des périphériques disposent d'une remise à zéro (Reset), qui est utilisée en cas de fonctionnement anormal. Le protocole USB prévoit ce cas avec la possibilité offerte à l'ordinateur hôte d'envoyer un signal USB de remise à zéro à chaud (USB Warm Reset) qui provoque une réinitialisation complète du périphérique. Cependant, les applications s'appuyant sur l'utilisation de cartes à puce peuvent se servir de la remise à zéro de la carte à puce dans le but de réinitialiser simplement la mémoire vive gérée par le microcontrôleur de ladite carte. Dans ce cas, la remise à zéro de l'interface de communication avec l'ordinateur hôte n'est pas nécessaire et génère une perte de temps. L'utilisation du signal «USB Warm Reset» n'est donc pas justifiée. D'autre part, ce signal «Reset» doit être complètement asynchrone, ce qui signifie qu'il doit pouvoir être pris en compte quel que soit l'état de la carte ou de la commande en cours de traitement, si une commande est effectivement en cours de traitement, ce qui justifie encore une fois l'emploi d'un lecteur de carte à puce dans les solutions proposées actuellement selon l'état de la technique qui effectue lui même la remise à zéro du microcontrôleur et de sa mémoire associée par la plage de contact connectée au plot de contact Reset du microcontrôleur.

Ensuite, quatre requêtes sont dédiées au traitement des commandes APDU. Il s'agit des requêtes SendAPDU(), GetData() et SendData().

La requête SendAPDU() permet d'envoyer à la carte l'entête d'une commande ISO APDU, c'est-à-dire les portions CLAsse, INStruction, paramètre P1, paramètre P2, et paramètre P3.

La requête GetData() permet à la fois de récupérer les données renvoyées par la carte dans le cadre d'une commande ISO de type 2 et de récupérer le mot d'état défini par la norme ISO7816, qui indique au monde extérieur l'issue de la commande précédemment envoyée lorsque l'exécution de la commande est terminée.

La requête SendData() permet d'envoyer des données en plus des paramètres de l'entête de la commande dans le cadre d'une commande ISO de type 3.

Enfin, la quatrième requête est une requête qui est utilisée pour éviter le déclenchement du mode de consommation réduite et pour gérer le déroulement des commandes APDU. Il s'agit de la requête IsReady(). Le traitement semi-automatique de la requête vendeur-spécifique IsReady() permet d'éviter le passage en faible consommation de courant durant l'exécution d'une commande ISO APDU. En effet, le temps de traitement d'une commande ISO APDU par la carte n'est pas prévisible. Or, le protocole USB prévoit un mode de faible consommation de courant lorsque le bus n'est pas utilisé pendant un certain temps, ce qui se produit si le temps de traitement de la commande ISO APDU est trop long. Cette requête évite ainsi le basculement dans ce mode pendant le traitement d'une commande ISO APDU tout en l'autorisant dans les autres cas. Plus précisément, elle permet de récupérer l'état du système d'exploitation de la carte ou de la commande en cours de traitement, si une commande est effectivement en cours de traitement. Elle est envoyée périodiquement, par exemple toutes les 5 ms, par le pilote 15 contenu dans l'ordinateur hôte pendant le traitement par la carte d'une commande ISO APDU. Elle peut être traitée par le moteur USB 333 contenu dans le microcontrôleur. C'est le cas, en particulier, lorsque le microcontrôleur est occupé (BUSY) ou en mutisme (MUTE) et ne peut donc pas répondre. Elle peut aussi être traitée par le système d'exploitation de la carte, en particulier lorsque celui-ci est libre et peut donc répondre.

L'ensemble de ces requêtes vendeur-spécifiques permet de reconstituer, en plus du fonctionnement classique de la carte à puce dans le mode ISO et du fonctionnement d'un périphérique standard USB, le comportement d'un lecteur de carte à puce associé.

Par ailleurs, une réponse dite OS_STATUS à la requête IsReady() a également été définie. Cette réponse est codée sur un octet dont les quatre premiers bits définissent l'état courant de la carte et les quatre derniers bits précisent cet état.

Ainsi, lorsque le bit 7 est à 1, cela signifie que la carte est dans un état de mutisme dit MUTE. Lorsque le bit 6 est à 1, cela signifie que le système d'exploitation de la carte est en cours de traitement et que par suite ce système n'est pas disponible pour un autre traitement. On dit alors que la carte est dans un état BUSY. Lorsque le bit 5 est à 1, cela signifie que le traitement de la commande antérieure reçue par la carte est terminé et que le système d'exploitation est prêt à envoyer un mot d'état SW1 SW2. On dit alors que la carte est dans l'état SWP pour Status Word Phase (phase de mot d'état). Lorsque le bit 4 est à 1, cela signifie que le système d'exploitation de la carte est prêt à envoyer ou à recevoir des données relatives à une commande antérieure. On dit que la carte est dans l'état DTP pour Data Transfer Phase (phase de transfert de données).

Par ailleurs, les bits 3, 2, 1 et 0 précisent l'état courant. Ils sont utiles par exemple dans le cas d'une commande très longue, pour éviter ce qu'on appelle un «Time Out», c'est-à-dire un dépassement du temps maximum prévu pour une commande. Dans ce cas, la valeur est incrémentée de manière cyclique. Elle reprend donc à 0h après la valeur Fh, ce qui permet au pilote contenu dans le PC de détecter une activité.

L'encapsulation pure, par le protocole USB, du protocole de communication défini dans les parties 3 et 4 de la norme ISO 7816, imposerait une perte de temps liée au fait que la carte ne peut transmettre des informations sur le bus USB que lorsqu'elle est sollicitée par l'ordinateur hôte et que certaines de ces

informations ne sont pas utiles dans le cadre de l'exécution d'une commande APDU. L'utilisation de la requête vendeur-spécifique IsReady() permet de réduire ce temps en indiquant au pilote de la carte non seulement l'état courant de la carte, mais aussi l'état courant de la commande, ce qui permet de supprimer l'étape d'octet de procédure (procedure byte) définie dans la norme ISO 7816-3.

La norme ISO 7816-3 prévoit la gestion d'un «Time-Out» si le système d'exploitation de la carte ne renvoie pas de données au bout d'un temps défini par la chaîne d'ATR. Pour les commandes ne pouvant être traitée dans ce temps, la norme prévoit également l'utilisation de l'octet 60h, qui constitue une valeur réservée, permettant d'indiquer que la carte est toujours en cours de traitement. L'envoi de cet octet par la carte a pour effet de réinitialiser le compteur chargé de déterminer le «Time-Out». La gestion de ce «Time-Out» peut être reproduite grâce à la valeur renvoyée à la requête IsReady().

A la figure 7, on a montré le déroulement d'une cession de communication avec une carte à puce selon l'invention. Cette figure montre, dans sa partie gauche, les traitements effectués par le moteur USB de la carte et, dans sa partie droite, les traitements effectués par le système d'exploitation de la carte.

En ce qui concerne les traitements effectués par le système d'exploitation de la carte, il s'agit en particulier des traitements suivants.

«Connexion de la carte» sur un port USB de l'ordinateur hôte. L'ordinateur hôte est alors informé de la connexion de la carte, qui constitue un nouveau périphérique USB. L'ordinateur alimente ensuite la carte, ce qui a pour effet de déclencher une remise à zéro de celle-ci. Cette remise à zéro comprend une remise à zéro de la mémoire RAM de la carte, de l'EBI 336, des registres 335 et du système de transission 334.

«Enumération et initialisation de l'ensemble des composants de la carte». L'énumération est une opération USB qui permet de rendre la carte opérationnelle, c'est-à-dire adressée et configurée. Une fois que la carte a fait l'objet d'une remise à zéro selon le traitement précédent, elle peut s'identifier

auprès de l'ordinateur hôte. C'est la phase dite d'énumération durant laquelle la carte envoie un certain nombre d'informations à l'ordinateur hôte sous forme de descripteurs. L'ordinateur hôte attribue alors une adresse à la carte et la configure. La carte apparaît alors prête à être utilisée.

«GetATR() reçu». Suite à l'étape précédente d'énumération et d'initialisation, la carte attend une requête vendeur-spécifique GetATR(). C'est d'ailleurs la seule requête vendeur-spécifique autorisée à ce stade.

«La carte renvoie la chaîne ATR». Lorsque la requête vendeur-spécifique GetATR() a été reçue, la carte renvoie la chaîne ATR. Ainsi, au niveau applicatif de l'ordinateur hôte, le "Reset" existant sur les cartes uniquement compatibles avec la norme ISO 7816 est simulé.

«OS_STATUS = 00h» : Le système d'exploitation de la carte se place dans une configuration dans laquelle il est prêt à traiter une commande ISO APDU en positionnant son octet d'état à 00h.

«SendAPDU() reçu» : Le système d'exploitation de la carte reçoit l'entête d'une commande APDU sous la forme d'une requête USB vendeur-spécifique.

«OS_STATUS = BUSY» : Le système d'exploitation de la carte se prépare à traiter l'entête de la commande APDU et devient donc indisponible. Pour indiquer cette indisponibilité au monde extérieur, en pratique à l'ordinateur hôte, ledit système d'exploitation met à jour son octet d'état en le positionnant à "BUSY". A ce stade, les requêtes provenant de l'ordinateur hôte sont traitées par le moteur USB de la carte.

«Traitement commande» : Le système d'exploitation de la carte traite l'entête de la commande APDU.

A ce stade, plusieurs cas peuvent se présenter.

Dans un premier cas, la commande est une commande APDU ISO de type 1, c'est-à-dire une commande APDU qui est représentée uniquement par son entête et dont l'exécution donne lieu à l'émission d'un mot d'état par la carte, ou une commande ISO de type 2 ou 3 en erreur, une commande ISO de type 2 étant une commande définie par son entête dont l'exécution donne lieu à

l'émission de données et d'un mot d'état par la carte et une commande ISO de type 3 étant une commande définie par son entête et des données et dont l'exécution donne lieu à l'émission d'un mot d'état par la carte. Dans ce cas, les étapes suivantes sont mises en œuvre.

«OS_STATUS = SWP» : Le système d'exploitation de la carte est prêt à renvoyer le mot d'état, qui est de nouveau disponible pour traiter les requêtes qui lui sont envoyées, et attend une requête IsReady() pour l'indiquer à l'ordinateur hôte. Cet octet d'état est ainsi mis à jour. Il prend la valeur "SWP".

«IsReady() reçu» : Le système d'exploitation de la carte reçoit ensuite la requête vendeur-spécifique IsReady(). Le rôle de cette requête est d'indiquer au monde extérieur l'état du système d'exploitation de la carte qui est "MUTE" ou "BUSY", ou alors, l'état de la commande ISO APDU en cours de traitement qui est "SWP" ou "DTP". Dans le cas présent, la réponse à cette requête est "SWP". Elle indique à l'ordinateur hôte qu'il doit envoyer une commande GetData() pour récupérer le mot d'état.

«Renvoie OS_STATUS» : le système d'exploitation de la carte renvoie son octet d'état à l'ordinateur hôte et attend une requête vendeur-spécifique GetData().

«GetData() reçu» : Après que la requête vendeur-spécifique lui ait été envoyée, le système d'exploitation de la carte reçoit la requête GetData(), qui est destinée à permettre à l'ordinateur de récupérer des données renvoyées par le système d'exploitation de la carte, telles que, dans le cas présent, le mot d'état.

«Renvoie le Mot d'Etat» : En réponse à cette requête GetData(), le système d'exploitation de la carte renvoie le mot d'état. Il se replace ensuite dans une configuration dans laquelle il est prêt à traiter une nouvelle commande ISO APDU et l'on revient alors à l'étape précitée «OS_STATUS = 00h».

Les commandes APDU ISO de types 2 et 3 ont la particularité de posséder une phase de données, de la carte vers l'hôte pour les commandes ISO2, et de l'hôte vers la carte pour les commandes ISO3. Dans ces deux cas, le système d'exploitation doit indiquer à l'ordinateur hôte qu'il est prêt pour la phase de données. Les étapes suivantes sont alors mises en œuvre.

«OS_STATUS = DTP» : Le système d'exploitation de la carte est prêt pour la phase de données de la commande ISO APDU. Il est donc de nouveau disponible pour traiter les requêtes qui lui sont envoyées et attend de recevoir une requête IsReady() pour indiquer cet état de disponibilité au monde extérieur. L'octet d'état est donc mis à jour. Il prend la valeur "DTP".

«IsReady() reçu» : Le système d'exploitation de la carte reçoit une requête vendeur-spécifique IsReady(). Le rôle de cette requête est d'indiquer à l'ordinateur hôte l'état du système d'exploitation de la carte qui est "MUTE" ou "BUSY", ou alors, l'état de la commande ISO APDU en cours de traitement qui est "SWP" ou "DTP". Dans le cas présent, la réponse à cette requête est "DTP". Dans un premier cas, cette réponse indique à l'ordinateur hôte qu'il doit envoyer une requête GetData(P3) pour récupérer les données qui constituent la réponse à la commande ISO APDU. Ces données comportent alors P3 octets, P3 étant l'un des paramètres de la commande ISO APDU. Dans un second cas, cette réponse indique à l'ordinateur hôte qu'il doit envoyer une requête SendData(P3) pour envoyer les données complémentaires de la commande ISO APDU. Ces données comportent alors P3 octets, P3 étant l'un des paramètres de la commande ISO APDU.

«Renvoie OS_STATUS» : Le système d'exploitation de la carte renvoie son octet d'état à l'ordinateur hôte et attend, soit une requête GetData(P3), soit une requête SendData(P3).

Deux cas peuvent alors se présenter.

Le premier cas est celui d'une commande APDU ISO2, dans le cas nominal où aucune erreur relative à l'entête de la commande ou dans le

contexte courant de la carte n'a été détectée. Le système d'exploitation de la carte attend alors une requête GetData(P3).

«GetData() reçu» : Le système d'exploitation de la carte reçoit la requête GetData(). Le rôle de cette requête est de récupérer des données renvoyées par le système d'exploitation, telles que, dans le cas présent, les données qui constituent la réponse à la commande ISO APDU de type 2.

«La carte renvoie les données» : Une fois la requête GetData() reçue, la carte renvoie les données qui constituent la réponse à la commande ISO APDU et se place dans une configuration où elle est prête à renvoyer le mot d'état et l'on revient alors à l'étape précitée «OS_STATUS = SWP».

Le second cas est celui d'une commande ISO APDU de type 3, dans le cas nominal où aucune erreur n'a été détectée dans l'entête de la commande ou dans le contexte courant de la carte. Le système d'exploitation attend alors une requête SendData(P3).

«SendData() reçu» : Le système d'exploitation de la carte reçoit la requête SendData(). Cette requête permet l'envoi de données complémentaires, qui sont nécessaires à l'exécution de la commande ISO APDU de type 3.

«La carte récupère les données» : Une fois la requête reçue, la carte récupère les données complémentaires de la commande ISO APDU de type 3 et se place dans une configuration qui lui permet de traiter le reste des données de la commande.

«OS_STATUS = BUSY» : Comme le système d'exploitation de la carte est en cours de traitement, il n'est plus capable de traiter les requêtes qui lui sont envoyées. Il indique cet état en positionnant l'octet d'état à "BUSY". Au cours de cette phase, c'est le moteur USB de la carte qui traite les requêtes envoyées par l'ordinateur hôte.

«Traitement de la commande» : Le système d'exploitation termine le traitement de la commande ISO APDU et se place dans une configuration où il

est prêt à renvoyer le mot d'état. On revient alors à l'étape précitée «OS_STATUS = SWP».

Un dernier cas traité par le système d'exploitation de la carte est celui d'une erreur grave survenant au cours de l'exécution d'une commande ISO APDU quelconque, par exemple suite à une attaque sécuritaire ou à une corruption de données. Dans ce cas, le système d'exploitation de la carte est placé en mutisme et on a l'étape suivante.

«OS_STATUS = MUTE» : Le système d'exploitation de la carte met à jour son octet d'état "MUTE" pour indiquer à l'ordinateur hôte qu'il est indisponible jusqu'à la prochaine requête DoReset() ou jusqu'à ce que la carte soit déconnectée. Durant cette phase, c'est le moteur USB qui traite les requêtes envoyées par l'ordinateur hôte.

En ce qui concerne les traitements effectués par le moteur USB, il s'agit en particulier des traitements suivants.

Lorsque l'OS est indisponible, c'est-à-dire dans les cas OS_STATUS = BUSY ou OS_STATUS = MUTE, les requêtes envoyées par l'ordinateur hôte sont traitées par le moteur USB. Par ailleurs, la requête DoReset() est toujours traitée par le moteur USB de manière à éviter toute intervention du système d'exploitation de la carte dans sa remise à zéro.

On distingue donc trois cas. Le premier, qui n'apparaît pas à la figure 7, correspond à l'ensemble des requêtes qui ne sont pas indiquées comme traitées par le moteur USB. Pour ces requêtes, le moteur USB se contente d'indiquer à l'ordinateur hôte qu'elles sont hors contexte.

Le second cas est celui de la requête DoReset() et les étapes suivantes sont suivies.

«DoReset() reçu» : Quel que soit l'état du système d'exploitation de la carte ou de la commande ISO APDU en cours de traitement, cette requête est toujours traitée par le moteur USB. Elle déclenche une remise à zéro de l'unité centrale de traitement de la carte associée à sa mémoire et seulement de cette

unité centrale associée à sa mémoire, puisque l'interface de communication USB formée par la configuration et l'adresse du périphérique restent intactes.

«Séquence de remise à zéro» : L'unité centrale de traitement de la carte et sa mémoire sont réinitialisés. Le système d'exploitation de la carte attend une requête GetATR(). La séquence de remise à zéro replace le système d'exploitation de la carte dans un état qui lui permet de traiter des requêtes qui lui sont envoyées.

Le troisième cas est celui de la requête IsReady(), lorsque le système d'exploitation de la carte est indisponible. Les étapes suivantes sont alors suivies.

«IsReady() reçu» : Le système d'exploitation de la carte reçoit la requête IsReady(). Le rôle de cette requête est d'indiquer à l'ordinateur hôte l'état "MUTE" ou "BUSY"du système d'exploitation de la carte, ou l'état "SWP" ou "DTP" de la commande ISO APDU en cours de traitement. Le système d'exploitation de la carte est indisponible "MUTE" ou "BUSY". Les autres cas sont traités par le système d'exploitation de la carte.

«Le moteur USB renvoie OS_STATUS» : Le moteur USB indique à l'ordinateur hôte l'état du système d'exploitation de la carte en renvoyant son octet d'état.

De cette manière, on reproduit le fonctionnement d'une carte ISO associé à son lecteur de carte à puce.

L'objet de la figure 7 ayant maintenant été expliqué, la suite de l'exposé constitue une explication de l'objet des figures 8A et 8B, 9A à 9D et 10A à 10D.

Pour une commande APDU de type 1 telle que montrée à la figure 8A, l'entête de la commande est suffisant pour exécuter entièrement la commande et la seule réponse du système d'exploitation de la carte est le mot d'état. Ainsi, selon le protocole ISO, la communication est divisée en au moins deux étapes. Dans une première étape, l'ordinateur envoie l'entête de la commande. Puis, dans une seconde étape, la carte envoie un octet 60h de remise à zéro du compteur chargé de définir le time-out ou un mot d'état SW1 SW2 (figure 8B).

Dans le cas où l'octet 60h est envoyé, des étapes subséquentes consistent en l'envoi d'autres octets 60h ou non, une dernière étant toujours l'envoi d'un mot d'état SW1 SW2. Par contre, selon l'invention, l'étape montrée à la figure 8B est supprimée. Elle est remplacée par la réponse de la carte à une commande IsReady() envoyée par ordinateur hôte.

Pour une commande APDU ISO de type 2, l'entête de la commande débute son exécution, mais la réponse du système d'exploitation de la carte est composée de données en plus du mot d'état. La communication est généralement divisée en quatre étapes. Dans une première étape conforme à la figure 9A, l'ordinateur envoie l'entête de la commande. La seconde étape est classiquement utilisée dans une procédure ISO. Dans cette étape, l'ordinateur reçoit l'octet de procédure 60h, INS ou SW1. Dans le cas où l'octet de procédure est 60h, on retourne au cas décrit ci-dessus jusqu'à réception de l'octet INS ou SW1. Une fois que l'octet de procédure INS ou SW1 a été reçu conformément à la figure 9B, on procède comme décrit ci-dessous en référence aux figure 9C et 9D. Toutefois, on notera que, dans l'invention, l'étape de la figure 9B est supprimée. En effet, elle est remplacée par la réponse de la carte à une commande IsReady() envoyée par l'ordinateur hôte. Si l'on se rapporte maintenant à la figure 9C, qui correspond au cas où l'octet de procédure reçu est INS, la carte renvoie les données. Finalement, l'ordinateur attend l'octet de procédure, jusqu'à ce qu'il soit SW1, conformément à la figure 9D. Si INS n'a pas été reçu mais SW1 a été reçu directement, on reçoit SW2 et la commande est terminée. Dans l'invention, les étapes montrées dans les figures 9C et 9D sont conservées sauf en ce qui concerne le cas visé dans la figure 9D dans lequel la carte renvoie l'octet de procédure 60h.

Pour une commande APDU de type 3, la procédure est identique à la procédure décrite ci-dessus en référence à au traitement d'une commande ISO de type 2 sauf en ce qui concerne le sens d'émission des données qui n'est plus de la carte vers l'ordinateur mais de l'ordinateur vers la carte.

REVENDICATIONS

1. Procédé de communication entre, d'une part, une station hôte telle q'un ordinateur personnel et, d'autre part, un objet portatif à microcontrôleur tel qu'une carte à puce, ledit objet portatif étant connecté, par un système de bus, à ladite station hôte, caractérisé en ce qu'il comporte une étape selon laquelle une requête spécifique est communiquée à l'objet portatif à microcontrôleur par la station hôte.
2. Procédé selon la revendication 1, caractérisé en ce que le système de bus est un système de bus série universel USB et en ce que la requête spécifique est communiquée à l'objet portatif à microcontrôleur selon le mode transfert de contrôle dudit système.
3. Procédé selon l'une des revendications 1 ou 2, caractérisé en ce que la requête spécifique est une requête spécifique qui assure une fonctionnalité d'un lecteur d'objet portatif à microcontrôleur.
4. Procédé selon l'une des revendications précédentes, caractérisé en ce que le microcontrôleur comporte un ensemble associant une unité centrale de traitement ainsi qu'une mémoire volatile et en ce que la requête spécifique est une requête (DoReset()) qui déclenche une remise à zéro de la mémoire volatile dudit ensemble.
5. Procédé selon l'une des revendications précédentes, caractérisé en ce que la requête spécifique est une requête (GetATR()) qui permet de récupérer une chaîne de réponse à une remise à zéro de l'objet portatif.
6. Procédé selon l'une des revendications précédentes, caractérisé en ce que la requête spécifique est une requête (SendAPDU()) qui permet à la station hôte d'envoyer à l'objet portatif une entête d'une commande.
7. Procédé selon l'une des revendications précédentes, caractérisée en ce que la requête spécifique est une requête spécifique (GetData()) qui permet à la station hôte de récupérer des données envoyées par l'objet portatif et de récupérer un mot d'état.

8. Procédé selon l'une des revendications précédentes, caractérisé en ce que la requête spécifique est une requête (SendData()) qui permet à la station hôte de communiquer des données à l'objet portatif.

9. Procédé selon l'une des revendications précédentes, caractérisé en ce que la requête spécifique est une requête (IsReady()) qui permet d'éviter un déclenchement, par la station hôte, d'un mode de consommation de courant électrique réduit chez l'objet portatif.

10. Procédé selon la revendication 9, caractérisé en ce que l'objet portatif communique une réponse (OS-STATUS) à la station hôte en réponse à la requête permettant le déclenchement, par ladite station, d'un mode de consommation de courant électrique réduit chez l'objet portatif, cette réponse étant codée de manière à définir un état courant de l'objet portatif.

11. Procédé selon la revendication 10, caractérisé en ce que l'état courant de l'objet portatif est un état de mutisme ou un état selon lequel la carte est en cours de traitement.

12. Procédé selon l'une des revendications précédentes, caractérisé en ce que l'objet portatif à microcontrôleur est une carte à microcontrôleur.

13. Procédé selon la revendication 12, caractérisé en ce que le microcontrôleur de la carte comporte une mémoire non volatile qui comprend un système d'exploitation susceptible de communiquer selon un protocole mettant en œuvre des commandes APDU telle que définies dans la norme ISO7816.

14. Objet portatif à microcontrôleur tel qu'une carte à puce, apte à communiquer avec une station hôte telle qu'un ordinateur personnel, par un système de bus connecté, d'une part, audit objet portatif et, d'autre part, à ladite station hôte, caractérisé en ce que l'objet portatif est apte à communiquer avec la station hôte, directement.

15. Objet portatif à microcontrôleur selon la revendication 14, caractérisé en ce qu'il est constitué par une carte à puce.

16. Objet portatif selon l'une des revendications 14 ou 15, caractérisé en ce que le système de bus est un système de bus USB.

17. Objet portatif selon l'une des revendications précédentes, caractérisé en ce qu'il comporte un ensemble associant une unité centrale de traitement et une mémoire non volatile embarquant un système d'exploitation apte à assurer une gestion de commandes APDU telle que définies dans la norme ISO7816.

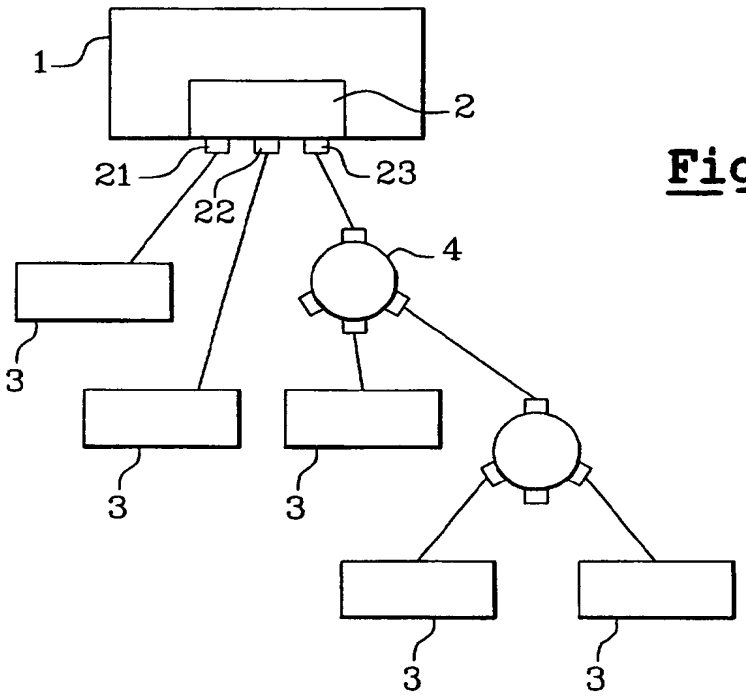


Fig. 1

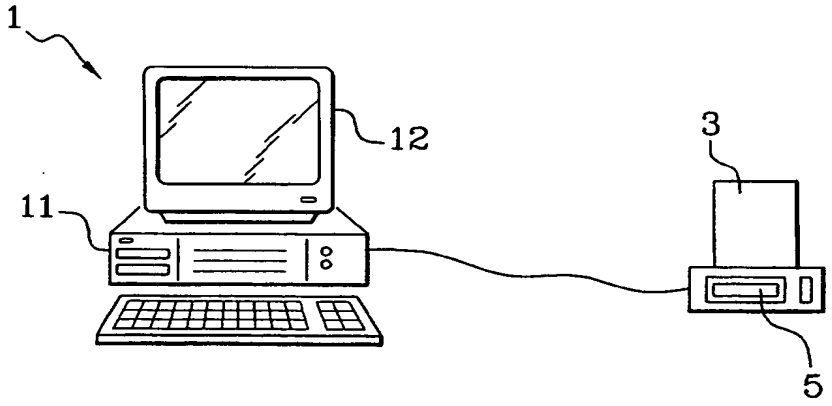


Fig. 2

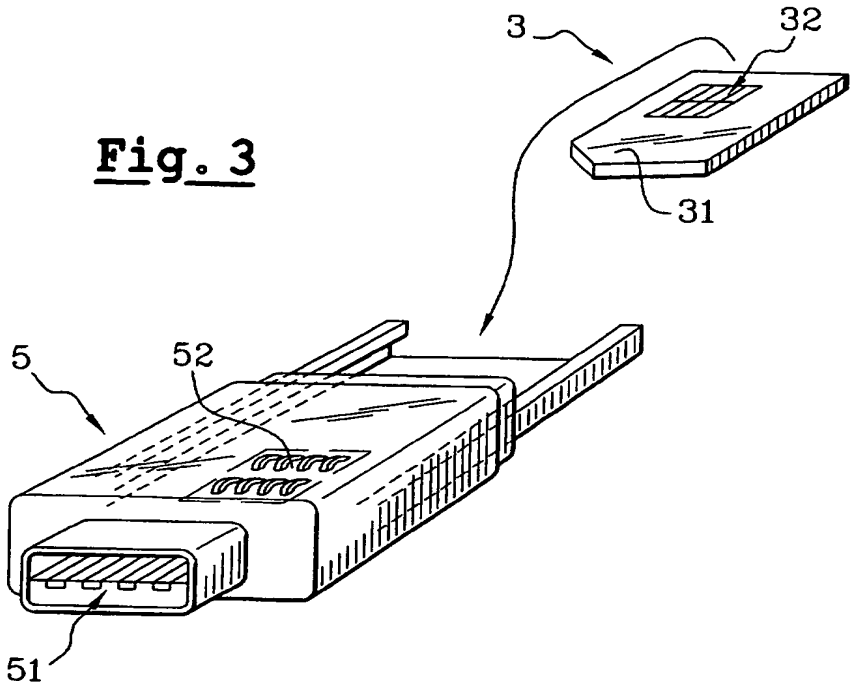


Fig. 3

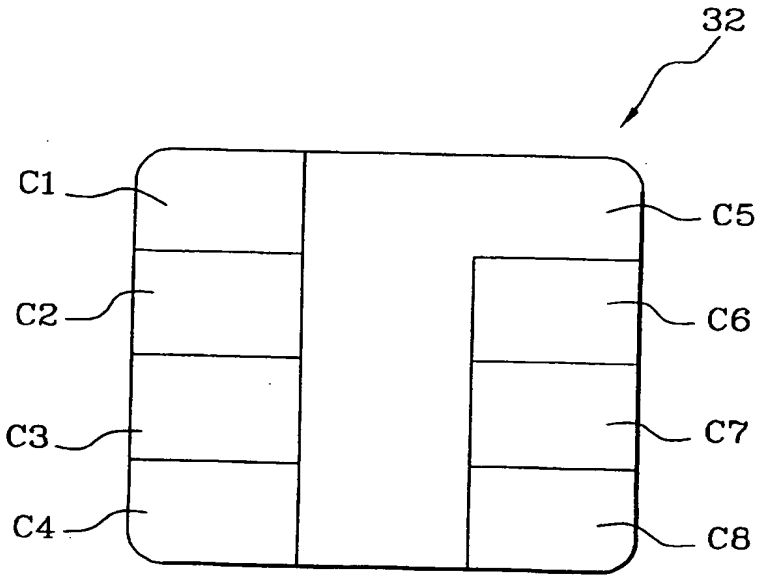


Fig. 4

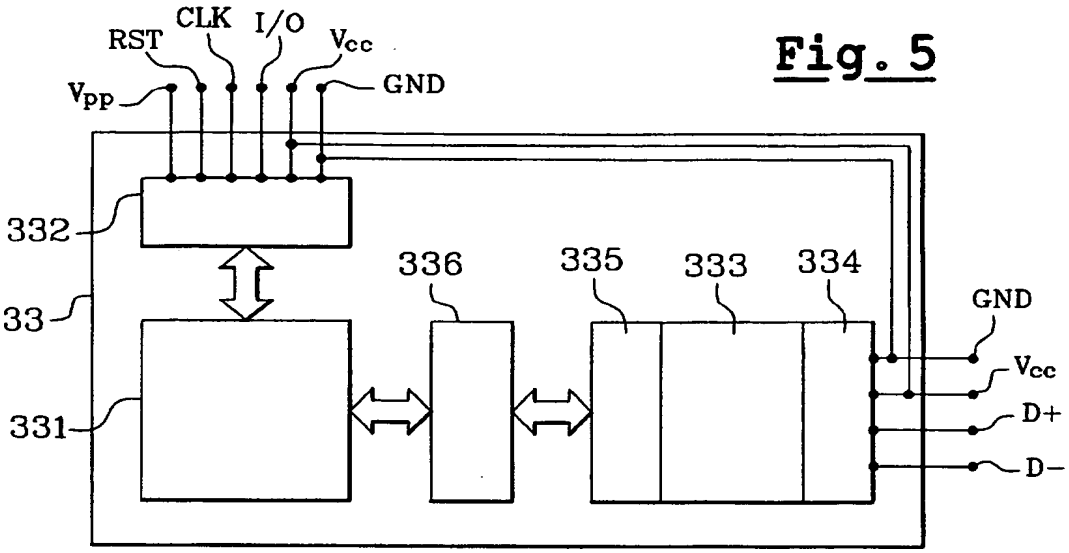

RST
CLK
I/O
Vcc
Vpp
GND
332
33
331
336
335
333
334
GND
Vcc
D+
D-

Fig. 5

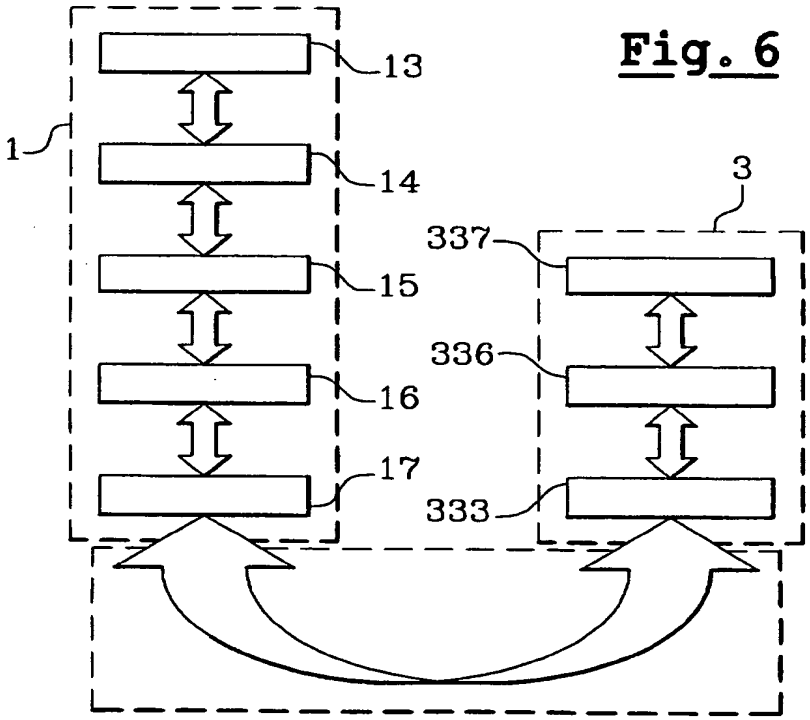

13
1
14
3
337
15
336
16
17
333

Fig. 6

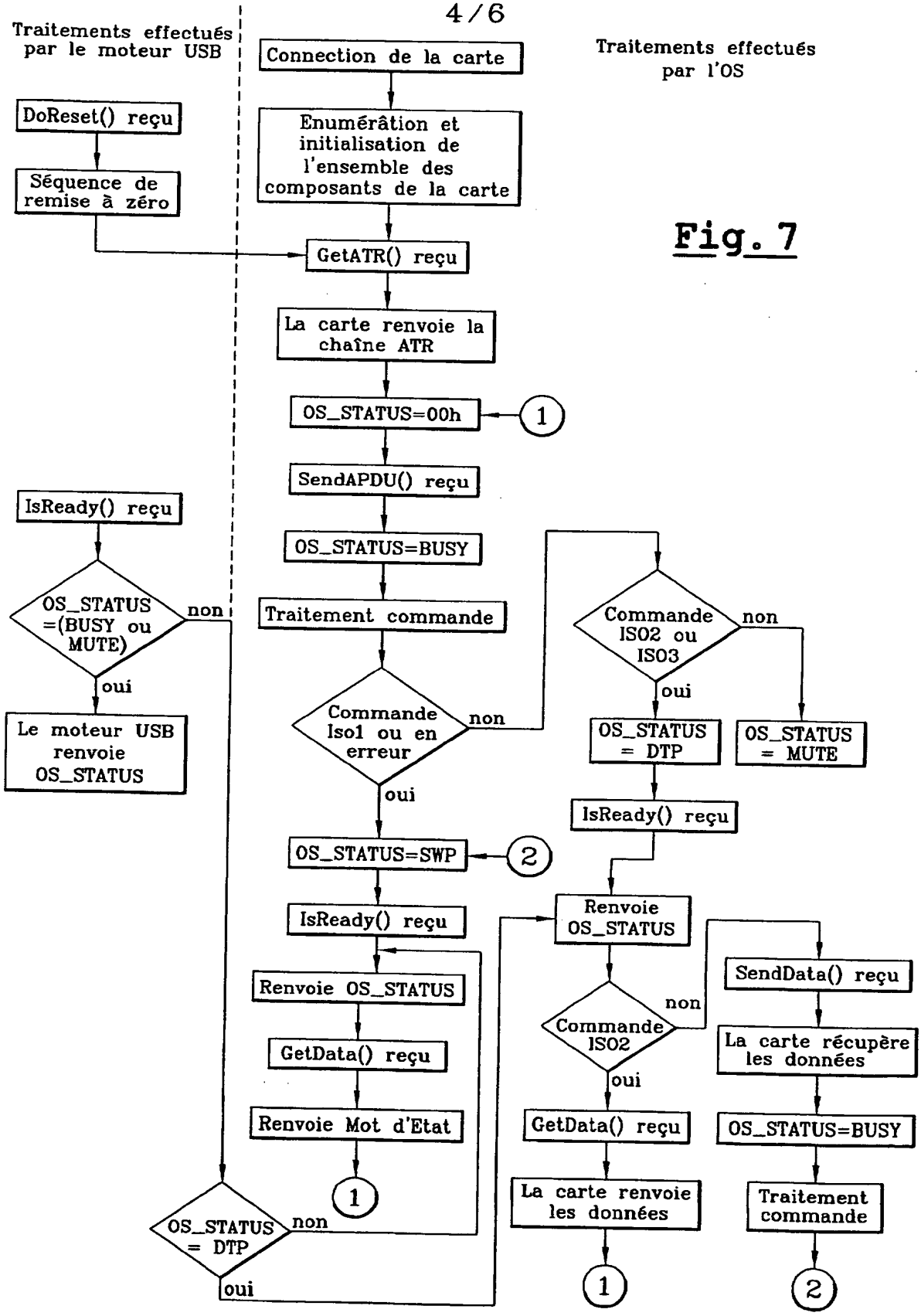

Traitements effectués par le moteur USB
Traitements effectués par l'OS
Connection de la carte
Enumérâtion et initialisation de l'ensemble des composants de la carte
DoReset() reçu
Séquence de remise à zéro
GetATR() reçu
La carte renvoie la chaîne ATR
OS_STATUS=00h
1
SendAPDU() reçu
IsReady() reçu
OS_STATUS =(BUSY ou MUTE)
non
oui
Le moteur USB renvoie OS_STATUS
OS_STATUS=BUSY
Traitement commande
Commande Iso1 ou en erreur
non
oui
Commande ISO2 ou ISO3
non
oui
OS_STATUS = DTP
OS_STATUS = MUTE
IsReady() reçu
OS_STATUS=SWP
2
IsReady() reçu
Renvoie OS_STATUS
GetData() reçu
Renvoie Mot d'Etat
1
OS_STATUS = DTP
non
oui
Renvoie OS_STATUS
Commande ISO2
non
oui
GetData() reçu
La carte renvoie les données
1
SendData() reçu
La carte récupère les données
OS_STATUS=BUSY
Traitement commande
2

Fig. 7

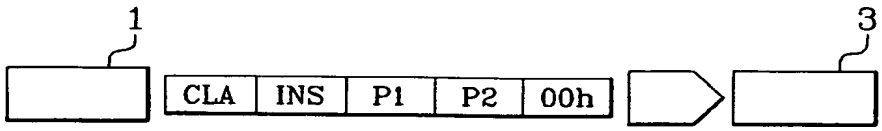


Fig. 8A

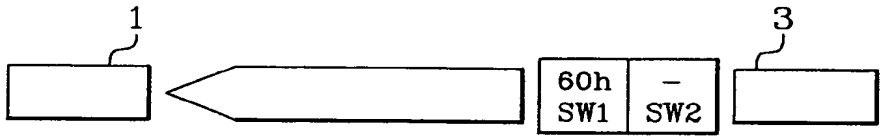


Fig. 8B

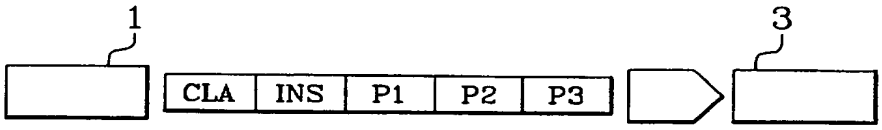


Fig. 9A

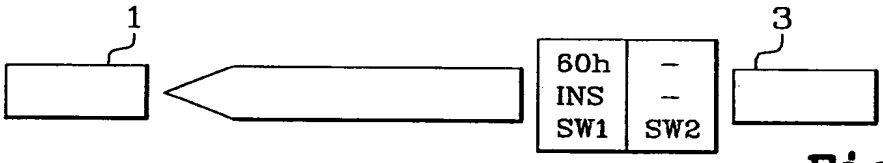


Fig. 9B

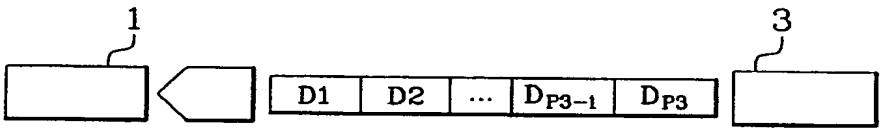


Fig. 9C

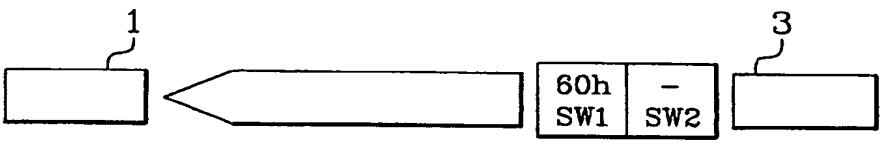


Fig. 9D

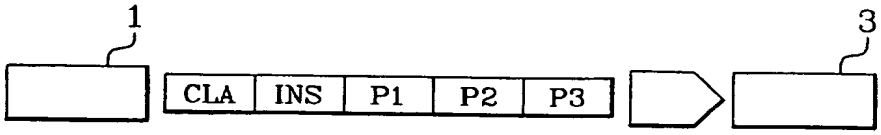


Fig. 10A

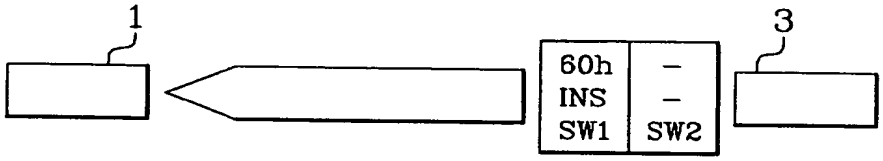


Fig. 10B

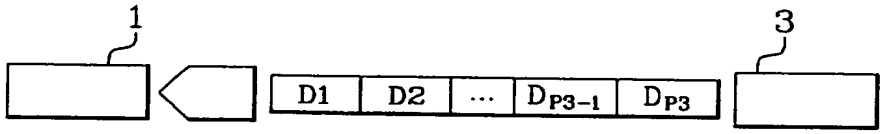


Fig. 10C

Fig. 10D

RÉPUBLIQUE FRANÇAISE

INPI
INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE

2806505

N° d'enregistrement national

FA 585787
FR 0003498

RAPPORT DE RECHERCHE PRÉLIMINAIRE

établi sur la base des dernières revendications déposées avant le commencement de la recherche

DOCUMENTS CONSIDÉRÉS COMME PERTINENTS		Revendication(s) concernée(s)	Classement attribué à l'invention par l'INPI
Catégorie	Citation du document avec indication, en cas de besoin, des parties pertinentes		
X A	FR 2 774 194 A (SCM SCHNEIDER MICROSYSTEME MIC) 30 juillet 1999 (1999-07-30) * le document en entier *	1-3, 14-17 4-13	G06K19/067
X A	US 5 767 844 A (STOYE DONALD A) 16 juin 1998 (1998-06-16) * le document en entier *	1,3,4, 14,15 5-13	
E	FR 2 783 336 A (SCHLUMBERGER IND SA) 17 mars 2000 (2000-03-17) * le document en entier *	1-3, 14-17	
			DOMAINES TECHNIQUES RECHERCHÉS (Int.Cl.7)
			G06K

Date d'achèvement de la recherche	Examinateur
4 décembre 2000	Degraeve, A

CATÉGORIE DES DOCUMENTS CITÉS

X : particulièrement pertinent à lui seul
Y : particulièrement pertinent en combinaison avec un autre document de la même catégorie
A : arrière-plan technologique
O : divulgation non-écrite
P : document intercalaire

T : théorie ou principe à la base de l'invention
E : document de brevet bénéficiant d'une date antérieure à la date de dépôt et qui n'a été publié qu'à cette date de dépôt ou qu'à une date postérieure.
D : cité dans la demande
L : cité pour d'autres raisons

& : membre de la même famille, document correspondant

EPO FORM 1503 12.99 (P04C14)